ITM Web of Conferences **13**, 01013 (2017) DOI: 10.1051/itmconf/20171301013
CMES2017

Modeling Routing Constructs to Represent Distributed Workflow Processes Using Extended Petri Nets

Mehmet Karay*
Final International University, Business Administrative, Toroslar Avenue, No:6, 99370, Catalkoy, Kyrenia, North Cyprus

Abstract.The contribution introduces explaining and modelling routing construct to represent distributed workflow processes using extended Petri nets with the new construction which is the interruption routing. Petri nets notation is used for representation of the four main routing constructs as well as for the workflow process. In this paper, four main routing constructs and the new construct interruption explained and modeledby using extended Petri nets.

1 Introduction

Workflow refers to a set of sequential tasks performed to achieve a pre-set goal in an organisation. In other words, workflow processes enable the flow of work among multiple participants. For contemporary business environments, Workflow is an essential component with its ability to build a well-equipped infra-structure for business processes by enhancing them with the supportive information and corresponding technology.

In order to manage workflow effectively, organizations started to employ technology in terms of computer software which is called Workflow Management System (WFMS). WFMS is defined as "A system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications" by Workflow Management Coalition [1]. WFMSs have recently been an important issue to investigate workflow processes in information systems of organizations. As a result of the challenges of globalization and the changes in economy along with the rapid improvements in technology, a growing competition has emerged within the market. So as to survive in this competitive environment, today's organizations must be time efficient, and cost effective. To address these requirements, it has been unavoidable for organizations to reconsider the way they organize workflow processes they use for workflow management systems.

* Corresponding author: mehmet.karay@final.edu.tr

As the need and importance of workflow systems has significantly increased for organizations over the last decades, it has been crucial to put serious efforts for their enhancement. It is a widely accepted point that Modelling is an efficient technique used in developing systems. Instead of working on the real phenomenon, it is more practical, cost effective and convenient to work on a model. There are a number of modeling approaches each coming with their weaknesses and strengths[11].

Among these approaches, Petri nets have been in use as a graphical modelling language for more than 50 years. They were first introduced by Dr. Carl Adam Petri in his Ph.D. thesis in 1962. Since then, they have been a successful modeling tool in computer science. Beneath the success of Petri nets lies their well-defined mathematical theory together with a well-organized bipartite graphical structure to represent active performance of systems. The theory of mathematics provides detailed modeling and analysis of the performance of system and the graphical structure makes it possible to visualize the performance of system.

W.M.P. van der Aalst, in his paper, provided three good reasons for using a Petri net based Workflow Management Systems (WFMS). The first reason was stated as the fact that business logic can be represented by a formal but also graphical language. The following reason was that in contrast with many other process modelling techniques, the state of case can be modelled explicitly in a Petri net. As the third reason, he stated that Petri nets are marked by the availability of many analysis techniques and this is a great asset in favour of a Petri net based WFMS.

However, Petri nets, in its ordinary form, are sometimes insufficient in some cases. That is because classical Petri nets do not represent a universal algorithmic system which means they cannot be used to create on arbitrary Turing machine. Therefore, they cannot be used to implement arbitrary algorithms of information processing, which is essential for the study of behaviour of dynamic system. This can be done only with the use of extended Petri nets that allow to express all fundamental properties of processes. These properties include transfer of data (in the form of move of tokens), time delay, control of move of tokens and data transformation. These properties are associated with firing of transitions in the extended Petri net. But, classical Petri nets express only move of abstract tokens from place to another places as a result of transition firing, and no time delay is associated with firing of transitions.For this reason, in this paper, extended Petri nets are used to model workflow processes.

2 A survey of a class of extended petri nets

As was stated in the introduction, general Petri nets, oriented from works of K. Petri in 1962, are not suitable for simulation of discrete-event systems. We give here a brief survey of a class of extended Petri nets that represent a complete algorithmic system and were used in this paper for simulation modeling. These nets are timed Petri nets, with attributed tokens and the abilities to control transfer of information, carried with token attributes, and to process this information during transition firing.

The structural elements of extended Petri nets are places, transitions and directed arcs. There are two types of places in these nets-simple and queue places.

A simple place or S-place (represented graphically as a circle) can hold one token at most, while a queue place or Q-place (depicted in the form of an oval) can accumulate an arbitrary number of tokens.

The minimal building blocks of extended Petri nets are elementary nets. Each elementary net consists of only one transition and a few incident input and output places. In the used extended Petri nets, there are five basic types of elementary nets, denoted by T, X, Y, I and G. Each type has its own structure and functionality. By connecting elementary nets with each other, one can create Petri nets of arbitrary size and complexity.

For the purpose of this paper, elementary nets of types T, Y, X and I were used. The following paragraphs contain brief descriptions of these types.

Elementary net of type T is shown in Figure 1. When its transition fires, the net merges tokens from all input places and creates new tokens in all output places. The transition can fire only if each input place contains a token and all simple output places (S-places) are empty. At the end of transition firing, the default data processing function, associated with this net, copies attributes of the token of place x_1 and assigns the values of these attributes to each new token in all output places y_1, y_2,…,y_n. If necessary, any desired data processing function can be performed on attributes of new tokens in output places. The net of this type actually implements logical operation AND on input places. Note that, in general, any non-zero time delay can be associated with the transition. This is true also for all other types of elementary nets. In addition, this elementary net, without input places, can be used as a token generator.

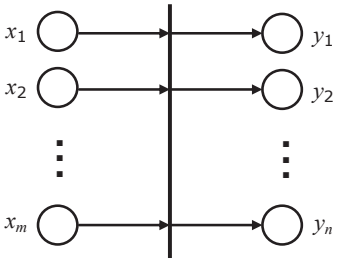


Fig. 1. Elementary net of type T.

Elementary net of type Y is given in Figure 2. As it is shown, the transition bar of this net has a short line segment directed to input places. Logically the net of this type implements the conditional multiplexing operation by selecting one token from one of the input places when the transition fires. As a result of transition firing, one token will be created in each output place. Values of attributes of the token, selected from an input place, are copied and assigned to corresponding attributes of new tokens created in all output places. On default, the token from the first non – empty input place is selected. But with an explicitly assigned input control function, the selection can be done from any other non – empty input place. Note that tokens in the remaining input places are not affected. Without input places, this elementary net behaves like a net of type T without input places.

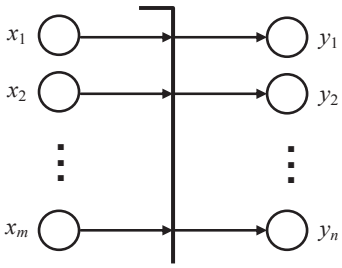


Fig. 2. Elementary net of type Y.

Elementary net of type X is presented in Figure 3. It provides routing of tokens from input places to one of the output places. The transition bar of this net has a short line segment directed to the output places. For firing of the transition here, it is necessary that each input place holds a token and the selected output place is empty. When the transition fires then, on default, attributes of input place x_1 are copied and their values are assigned to attributes of a new token created in one of the output places.

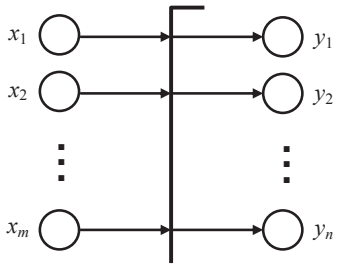


Fig. 3. Elementary net of type X.

Elementary net of type I, shown in figure 4, provides a possibility to interrupt the firing interval of transition by an external event. As shown in the figure, this net has two input places x1 and x2 and two output places y1 and y2. Place x1 is the main input, while place x2 serves as an interrupting place. Correspondingly, y1 and y2 are outputs for the main input and for the interrupting input, respectively. The arrow symbol on the transition bar is directed from the interrupting input to the main input.

There are two main scenarios in the operation of this elementary net. First, if there is a token in place x1, but place x2 is empty, the net works as usual net of type T with two incident places x1 and y1. Second, if, during started firing of transition, a token appears in place x2, then the ongoing firing instantly stops, the token from place x1 goes to place y1 with its attributes, and the interrupting token from place x2 goes, with its attributes and zero delay, to place y2. After the interruption, the tokens in place y1 and y2 can be treated separately in subsequent parts of the model. Usually, the interruptingtoken will initiate some interruption procedure by this same net, while the interrupted token will wait until the net completes processing the interruption event.

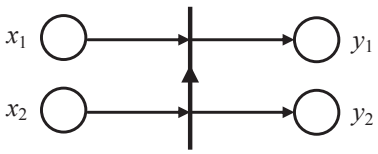


Fig. 4. Elementary net of type I.

Detailed description of the elementary nets can be found in [5].

For the description of models in terms of extended Petri nets there is the Model Description Language (MDL). The MDL is an extension of object Pascal, with added features for the description of elementary nets, token attributes and initialized data. There is also Modelling Control Language (MCL) that is used to parameterize the compiled models and control their runs on the computer. All these capabilities are implemented in the simulation system Winsim. Its software, user guide and many examples of solved models are available on the CD attachment of [5].

3 Concepts of a workflow process

A workflow process can be examined based on three main concepts. These are case, task, and condition.

The case is the situation in which the need for a workflow management is emerged. Workflows are founded by different cases. The execution of each individual work is performed for a specific case. To illustrate, a healthcare data analysis or an insurance claim both represent different cases. Dealing with cases in the most successful way is the primary objective of workflow management. A workflow process always handles similar cases.

Throughout a workflow process, the case is handled by executing sequential tasks. A task is a piece of work to be done in a certain time interval. The workflow process definition includes the specification of tasks to be carried out and also their order of execution depending on a set of conditions. Therefore the order of the tasks is pre-determined and directed by the set of conditions. A condition holds (true) or does not hold (false). Each task has pre- and post-conditions: the preconditions should hold before the task is executed, and the post conditions should hold after the task is executed [Aalst, 2004].

When Petri nets are used to model a workflow process, tasks are modeled by transitions, conditions are modeled by places, and cases are modeled by tokens.

4 Basic types of workflow constructs

A case is handled throughout a process. The paths that regulate the flow of a process are called routes. The routing of cases is one of the main issues of a workflow process as it determines which process should be done first and which one should go next. These routes are constructed in five different ways. These are sequential, parallel, conditional, iterative, and interruption construct. The first four

are applied in the previous studies of modeling workflow processes. In this thesis, as an innovation, interruption construct is also employed for modeling of a workflow process for the first time.

Sequential construct is used when the tasks need to be performed one after another. The tasks which are ordered sequentially are interrelated to each other. One task has to be completed in order to move on the next one. An example of a sequential construct is shown in Figure 5(a).

Differing from sequential routing, parallel construct enables the process to cope with more than one tasks at the same time or in any order. Figure 5(b) shows an example of parallel construct.

Some cases are subject to relevance and they might need to be handled in different ways under different circumstances. The alternative route to be selected varies depending on a specific condition. To model this type of process, conditional routing is used. An example of a conditional construct can be found in Figure 5(c).

In some cases, it is required to repeat the execution of a task. Multiple execution of a particular task is managed by iterative routing. For instance, this type of routing is used when a task requires to be repeated until a subsequent test gives positive results. Figure 5(d) illustrates an example of iterative construct.

The last and the most important form of workflow construct in this thesis is interruption. Interrupts are widely used in information systems to temporarily stop a running process and allow a process with a higher priority to run instead. This type of workflow construction is needed to enable the process to perform a more urgent service. In this case, the interrupted process will continue running after the termination of the process with the higher priority. Figure 5(e) shows an example of interruption construct.

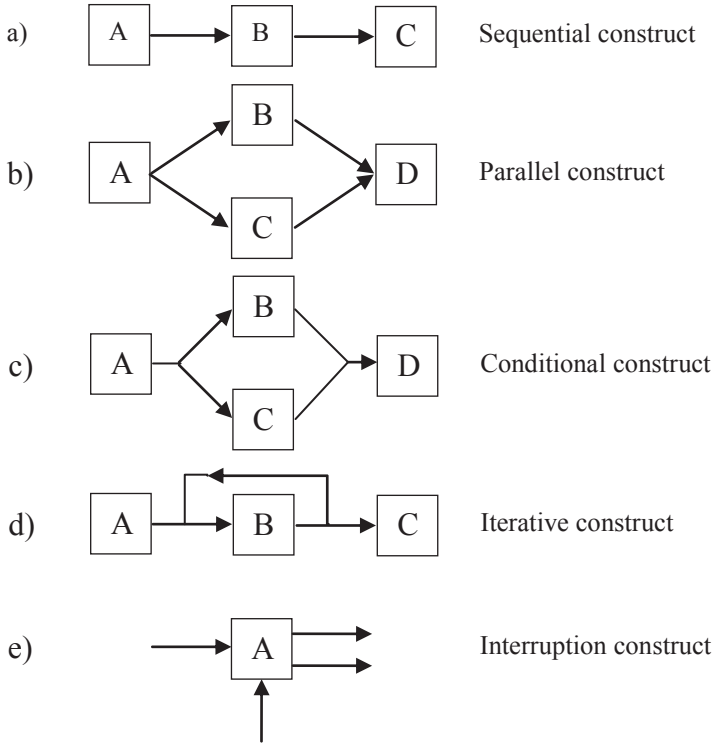


Fig. 5. Workflow constructs

5 The use of basic types of workflow routing constructs to describe a set of interacting distributed workflow information systems

In this section, an example set of interacting distributed workflow information systems is described by using basic types of workflow routing constructs. The example considers a travel agency organizing a trip (Aalst, [2000]). In this case, there are a number of tasks for the travel agency to execute.

In the first place is the registration of the customer. After that, an employee looks for chances to contact the customer. The next step is to reach the customer to learn if she or he is still interested in the trip or if she or he would like to get information about some other alternatives. So far these tasks should be modeled by using sequential routing construct as all the tasks are executed one after another.

From this point on, the process has three possibilities. The first one is when the customer is not interested anymore. The second is when the customer would like to see more alternatives, and the last one is when customer selects an opportunity. If the customer isn't interested in the trip at all, the process comes to an end as a consequence. If the customer wants to be informed of other alternatives, then the

employee starts to search for the required information to contact the customer again later. So, the process is repeated and this is modeled by iterative routing construction. On the other hand, if the customer decides to book a trip, then the system both books the trip and starts to prepare one or two types of insurance on request at the same time. As these processes are done simultaneously, they are modeled by parallel routing construction.

There are two types of insurance provided by the agency; trip cancellation insurance and baggage loss insurance. The customer can only choose to get the trip cancellation or only the baggage loss as well as she or he can choose both. Another option for the customer is not to choose any types of insurance. This decision making process is modeled by conditional routing construction.

With the above example, the use of four workflow routing constructs is described in process. The fifth routing construct which is the most significant one for this thesis, named as interruption, is investigated by the researchers in the literature [10].

6 Petri nets as an efficient formal tool to represent and investigate systems of distributed workflow processes

In this section, it is briefly explained why Petri nets are considered to be a desirable theoretical framework for the formal description, modeling and analysis of distributed systems.

To start with, it is convenient to design models in graphical form with Pet nets as Petri nets themselves are graphs consisting of a limited set of types of structural and dynamic elements. Also, several software packages along with their broad range of analyzing procedures can be found to support the design of Petri nets.

Next, using Petri nets gives the advantage of being able to represent various resources of distributed systems such as queues and data bases. The availability and the quantity of the corresponding resource is modeled by the existence of one or more tokens in a place.

In Petri nets, firing of transitions is used to model events related to these transitions. A process considered as a sequence of events is represented by the corresponding sequence of firing transitions of the net. Many transitions are allowed to fire in parallel which enables modeling the existence of many parallel processes in a system together with representing different types of process synchronization.

Another benefit of Petri nets is that they enable detecting deadlocks, proving the properties of safeness, boundedness and liveness, and determining the possibility of reachability of some *target state* from a given *initial state*. The reachability analysis represents one of the most important tasks in the study of discrete-event systems and, it is possible to solve that formally with the help of Petri-net-based models.

Petri nets are also practical to use with large and complex distributed systems as it is possible to use them for modular and hierarchical modeling.

The last but not the least, Petri nets have a number of extensions such as Petri nets with inhibitor arcs, timed Petri nets, stochastic Petri nets, colored Petri nets, predicate/transition Petri nets which enable qualitative and quantitative

analyses. These are called extended Petri nets. In some types of extended Petri nets, tokens have attributes which controls the movements of token in the net and in the data transformation during a transition firing process. These types of extended Petri nets are universal algorithmic systems which are suitable to be used for simulation modeling of information systems.

The formal definition and different characteristics of the formal analysis of Petri-net based models are given in more detail in the [10].

7 Mapping of basic types of workflow routing constructs into extended petri nets

The basic type of workflow routing constructs include sequential, parallel, conditional, iterative, and interruption routings which are previously described in Chapter 2. In this section, mapping of these workflow routing constructs into extended Petri nets is described. Building blocks are used to model workflow routing constructs.

Aallst (2004) explained and applied the first four constructs in his paper.

Sequential routing: Causal relationships among transitions (tasks) are dealt with sequential routing. Considering two tasks *A* and *B* and assuming that task *B* is executed after the end of task *A*, then it can be said that *A* and *B* are executed sequentially. Table 1(a) illustrates how the sequential routing can be modeled by adding places to model causal relations among transitions. S2models causal relation between task *A* and task *B*.

Parallel routing: This routing construct is applied in cases where the order of execution is less strict. When two tasks B and C need to be executed but the order of execution is arbitrary, elementary net of type T is used as a building block. Table 1(b) involves an example of such a parallel routing where tasks *B* and *C* are executed in parallel.

Conditional routing: This routing construct is utilized to let routing vary between cases. This makes the routing of a case depended on the workflow attributes of a case, the behaviour of the environment, or the workload of the organization. For the modeling of two or more alternatives, two building blocks are used. These are elementary nets of Type X and Y. This kind of routing is shown in Table 1(c) where task A is has the chance to be followed by either task B or task C. The choice is made between task B and task C. Followed by the execution of one of these two tasks, task D is executed.

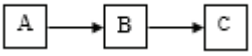
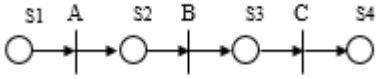
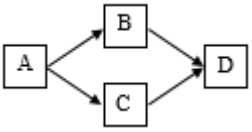
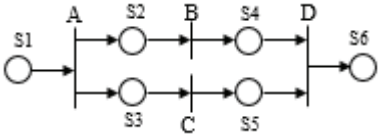
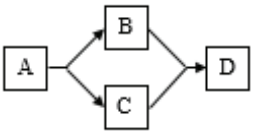
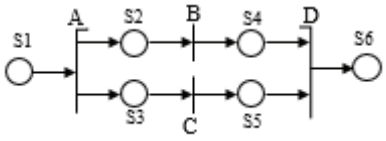
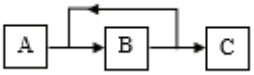
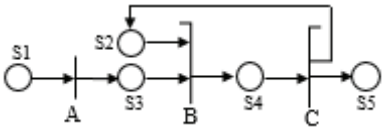
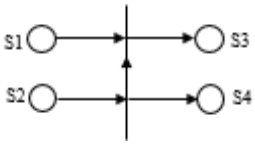
Iteration routing: This routing construct is employed to allow repeated execution of a particular task.As building blocks elementary nets of type X and Y are used for modeling. As shown in Table 1(d), task *B* may be executed one or more times where task *C* is the control task.

Interruption routing: In addition to those pre-applied constructs by Aallst(2004), this study also employs interruption routing which provides a very important possibility for an external event to interrupt an activity during its firing transition. To model this type of routing construct, elementary net of type I is used as shown in Table 1(e).

Aallst(2004) modeled workflow management systems with classical Petri nets by using sequential, parallel, conditional and iterative routing constructs. This

study takes another step with extended petri nets by adding the *interruption* feature which elementary net of type I makes possible.

Table 1. Basic types of workflow processes as given by Aallst(2004) and in terms of extended Petri nets

Types of the workflow process	By Aallst (2004)	With the extended Petri nets
a)Sequential		
b)Parallel		
c)Conditional		
d)Iterative		
e)Interruption	N/a	

8 Conclusion

Petri nets are very powerful modeling tool with a convenient graphical representation. Their ability to represent complex systems makes them useful for performance analysis. Extended Petri nets refer to a more featured version of Petri nets which allow transfer of data, time delay, control of move of tokens and data transformation. These properties are associated with firing of transitions in the extended Petri net. Classical Petri nets express only the move of abstract tokens from place to another places as a result of transition firing, and no time delay is associated with firing of transitions.

Modeling workflow processes using extended Petri nets allows application of different analysis methods that can be used to examine the behaviour of the process and to compute its performance measures. Transforming Extended Petri net model into graph model allows to express all aspects of control and data transformation in the models of information systems. The extended Petri nets include all basic constructs of work-flow processes proposed by W. Aalst and a new construct for dealing with the interruption of processes. There is a simulation system Winsim for implementation and running simulation models in terms of extended Petri nets. Developed workflow routing constructs in terms of extended Petri nets was used and investigated theoretically and in simulation experiments the model of a queuing system with relative and absolute priorities and developed in terms of extended Petri nets and investigated theoretically and in simulation experiments the model of a cell of flexible manufacturing cell.

References

1. Workflow Management Coalition: Workflow Management Coalition – Terminology &Glossary. Technical Report. Document Number WFMC-TC-1011 (1999).
2. W. M. P. van der Aalst, K. M van Hee: Workflow Management – Models, Methods, and Systems, The MIT Press, Cambridge, Massachusetts, London, England, 2002.
3. Hollingsworth, D.: Workflow Reference Model. Technical Report. The WorkflowManagement Coalition, Document Number WFMC-TC-1003 (1995).
4. Desrochers, A., and R. Ai-Jaar. 1995. Applications of Petri Nets in Manufacturing Systems: Modeling, Control and Performance Analysis. IEEE Press.
5. A. Kostin and Ilushechkina L, "Modeling and simulation of distributed systems", World Scientific Publ. Co., 2010.
6. T. Murata, Petri nets: Properties, Analysis and Applications, Proc. IEEE, vol. 77, no. 4, pp. 541 – 580, 1989.
7. Kostin, A.E., Models and Algorithms for Organization of Distributed Data Processing in Information Systems, Diss. DSc , Moscow Institute of Electronic Technology (Technical University), 1989 (in Russian).
8. Y. Fanaeian and A. Kostin, "Simulated Study of an Anycast – Based Rotuing Method for Wireless Sensor Networks with the use of Petri nets", International Journal of Science and Advanced Technology, vol. 3, no. 4, pp.18 – 27, 2013.
9. Lu, R., Sadiq, S.: A Survey of Comparative Business Process Modeling Approaches. BIS 2007, 82-94 (2007).
10. M. Karay and A. Kostin, "Using Extended Petri Nets for Modeling and Simulation of Queuing Systems with Priorities", International Journal of Science and Advanced Technology, vol. 4, no. 7, pp. 1-6, 2014.